Amendments to the Specification:

Please replace paragraphs [0024], [0026], [0036], [0045] and [0047] with the following amended paragraph:

[0024] Figure 9 is a simplified schematic diagram illustrating a system for displaying a user interface in which a draw ~~drawer~~ manager is responsible for updating a display rather than an application in accordance with one embodiment of the invention.

[0026] Figure 11 is a simplified schematic diagram of status bar which further exemplifies the advantages of having a draw ~~drawer~~ manager as a middle man in one embodiment of the invention.

[0036] Carlets 132 of Figure 4, have access to each layer above and including OS layer 122. Application program interface (API) layer 130 is the layer that carlets use to communicate with the applications to be processed by the telematics control unit (TCU) ~~IPS~~. Service provider interface (SPI) layer 128 is a private interface that managers have among each other. One skilled in the art will appreciate that OSGI layer 126 provides a framework upon which applications can run. Additional functionality over and above the JVM, such as lifecycle management is provided by OSGI layer 126. It should be appreciated that the open services gateway initiative is a cross industry working group defining a set of open APIs for a service gateway for a telematics systems. These APIs consist of a set of core framework APIs. In order to deploy services and their implementations OSGI defines a packaging unit called a service bundle. A service bundle is a Java Archive (JAR) file containing a set of

service definitions along with their corresponding implementation. Both infrastructure services and carlets are deployed as service bundles. Some of the functionality for arbitrating, controlling and managing devices and resources, e.g., speakers, cell phones, etc., by OSGI layer 126. However, one skilled in the art will appreciate that a separate arbitration service may also be required. As used herein, a carlet is a Java application. For each function or task to be processed on the client side or between the client and server sides, a carlet is invoked to manage the operation. In this manner, carlets can be independently written, tested, and launched for use on a telematics system. By way of example, a carlet can be written to control or monitor the activity of automobile components (e.g., tires, engine oil, wiper activity, steering tightness, maintenance recommendations, air bag control, transmission control, etc.), and to control or monitor applications to be processed by the telematics control unit (TCU) and interacted with using the on-board automobile monitor. As such, specialized carlets can be written to control the audio system, entertainment modules (e.g., such as on-line games or movies), voice recognition, telecommunications, email communications (text and voice driven), etc. Accordingly, the type of carlets that can be written is unlimited. Carlets may be pre-installed or downloaded from a sever. A carlet may or may not have an API which may be invoked by other carlets and it may or it may not have running threads of its own.

[0045] Figure 9 is a simplified schematic diagram illustrating a system for displaying a user interface in which a draw manager is responsible for updating a display rather than an application in accordance with one embodiment of the invention. Here, application 550 writes to application buffer 552, which in turn transmits data to draw manager 554. It should

be appreciated that draw manager 554 includes memory and code. In one embodiment, draw ~~drawer~~ manager 554 includes a display buffer such as the display buffer described with reference to Figures 6 and 7. Draw manager 554 is in communication with hardware 556. One skilled in the art will appreciate that hardware 556 may include a graphics processor. Hardware 556 is in communication with display screen 558. As can be seen through Figure 9, application 550 does not send data to the drawing code directly. Application 550 is configured to update the draw manager at a certain rate. The draw manager then takes on the task of updating the view on display screen 558. For example, if part of the view being displayed on display screen 558 is obscured, then un-obscured, draw manager 554 performs the re-draw and not the application 550. As a result of the application writing as often as the application desires, and the system writing as often as the system desires, the system performs the optimization. Thus, draw manager 554 may be configured to be optimized depending on the particular operating system and hardware 556 incorporated in the computing device in accordance with one embodiment of the invention.

[0047] Figure 11 is a simplified schematic diagram of status bar which further exemplifies the advantages of having a draw manager as a middle man in one embodiment of the invention. Here, the status bar is constantly being updated as some process is progressing. For example, the draw ~~drawer~~ manager is updating a display every 5 millisecond but an application may be performing an update every 1 millisecond. Thus, the draw ~~drawer~~ manager is configured to optimize the display based upon the hardware and operating system for the computing environment in which the draw manager resides.